VideoExcel & Fovea Image processor API Protocol

Revision History

Version	Author	Modification	Page	Date
1.00	R.Taylor	Initial Release		15 January 2010
1.01	R.Taylor	Updated API calls to match final specification		17 December 2010
1.33	D. Bartholdtsen	Added FoveaHD API calls Corrected Identity Reply message		19 June 2012
1.40	D. Bartholdtsen	Added FoveaHD API calls		18 November 2013
1.41	D. Bartholdtsen	Changed Fovea-HD to Fovea		11 December 2014

This manual details the protocol used to remotely control your Video Excel image processor. Where pseudo code can clarify the protocol, it has been included.

If you have any queries relating to this or any other product supplied by Calibre please visit our web site www.calibreuk.com***.***

For technical support please e-mail techsupport@calibreuk.com ***or send your queries by fax to (44) 1274 730960, for the attention of our Technical Support Department.***

Calibre UK Ltd
Cornwall House, Cornwall Terrace
Bradford, West Yorkshire
BD8 7JS, England

Telephone	***+44 (0)1274 394125***
Fax	***+ 44 (0)1274 730960***
Email	***techsupport@calibreuk.com***
Web-site	***www.calibreuk.com***

Contents

1 Introduction

This manual detail the protocol used to remotely control your Video Excel image processor, where pseudo code clarifies the protocol it has been included.

2 Communication Ports

The AV API protocol provides remote control functionality of units via RS232 and ethernet.

2.1 RS232

The RS232 port is a 9-pin female connector. It is a three wire connection: RX, TX and ground.

The unit has the following serial port settings:

baud rate	9600
parity	none
data bits	8
stop bits	1

2.2 Ethernet

The ethernet connector is a RJ45 and runs at 10Mb/s.

To connect using ethernet the following settings are required.

IP Address	User Selectable / from DHCP
Subnet Mask/Extended Network Prefix	User Selectable / from DHCP
Port Number	30001

The user may choose to obtain the IP Address, subnet mask and extended network prefix from DHCP or set them statically using the front panel (or GUI when available). The factory default is to obtain them from the DHCP server.

2.3 Timing Constraints

After sending the request packet and before receiving the reply packet it is necessary to have a delay. The following table provide a list of delays for specific parameters. For parameters not in the table the delay should be 10 milliseconds. Failing observe this wait may have adverse effects on the unit, causing lock-ups and data corruption requiring at least a power cycle and possibly a factory reset.

Parameter Identifier	Delay
avPar_pan	250 ms
avPar_tilt	250 ms
avPar_zoomX	250 ms
avPar_zoomY	250 ms
avPar_outputWndEdgeLeft	250 ms
avPar_outputWndEdgeRight	250 ms
avPar_outputWndEdgeTop	250 ms
avPar_outputWndEdgeBottom	250 ms
avPar_aspectRatioWidth	10 ms
avPar_aspectRatioHeight	10 ms
avPar_outputPictWarpTLX	250 ms
avPar_outputPictWarpTLY	250 ms
avPar_outputPictWarpTRX	250 ms
avPar_outputPictWarpTRY	250 ms
avPar_outputPictWarpBLX	250 ms
avPar_outputPictWarpBLY	250 ms
avPar_outputPictWarpBRX	250 ms
avPar_outputPictWarpBRY	250 ms
avPar_saturation	50 ms
avPar_hue	50 ms
avPar_chCurr_pipPosX	50 ms

avPar_chCurr_pipPosY	50 ms
avPar_chCurr_pipSizeXY	50 ms
avPar_outputDisplay	* 300 ms
avPar_outputGammaMode	250 ms
avPar_pictureFormat	150 ms

* This parameter has proved to be particularly troublesome and unpredictable with regard to the delay. It is recommended that special attention be paid and err on the side of caution.

3 API Functions

All communications to the unit have fixed request packet size of 16 bytes. There are different reply packets for the class of function. All are fixed length except when a string is returned, these are intrinsically variable in size. See 3.2.2 Parameter Reply Packet format for details.

Some functions will cause the unit to reboot and this will break the communication link. The user interface designer will need to consider this scenario as part of the user interface design or document accordingly.

3.1 Identity

The identity function provides the ability to test that the unit being controlled understands the AV API protocol that this document refers.

3.1.1 Identity Request Packet format

The identity packet is filled with ASCII uppercase ‘A’s.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’	‘A’

3.1.2 Identity Reply Packet

The reply packet is variable in size depending on the reply. For Ethernet the reply packet may be padded for performance purposes. The host should expect to receive at least 1024 bytes and allocate a buffer large enough.

A unit understanding this protocol will return the following ASCII string (here: Fovea):

1	2	3	4	5	6	7	8
‘V’	‘X’	‘L’	‘5’	‘0’	‘0’	‘H’	‘D’

(This refers to the protocol software and not the physical units’ product or model).

It is envisaged that protocol revision will be made available also, to enable the pairing of firmware with remote control software.

3.2 Parameter Functions

The AV API parameter functions provide a means to query and control the unit remotely.

3.2.1 Parameter Request Packet format

The parameter request packet takes the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
‘A’	‘P’	fun	paramID				attribute		value						‘A’

Where

‘A’, ‘P’	are literal ASCII characters.
fun	is the function identifier.
paramID	is the parameter identifier.
attribute	is the attribute identifier.
value	is a value.

The fun, paramID, attribute and value values are numeric values (with the exception of set string) using Base 64 (see 3.4 Base 64 Encoding) and must be fully padded. Base 64 is used as a compromise between compressing numeric values whilst still using user readable characters.

3.2.1.1 Function

The function identifier specifies the action or request to be performed.

Identifier	Value	Base 64	Description
avrFunc_query	0	A	Query the API functionality for parameter
avrFunc_get	1	B	Get a value
avrFunc_set	2	C	Set a value
avrFunc_gets	3	D	Get a string
avrFunc_sets	4	E	Set a string

The functions are detailed individually in their own sections below.

3.2.1.2 Parameter Identifier

The parameter identifier specifies this action or request is related to. There are different types of parameters and as such have a different set of functionality available.

Identifier	Value	Base 64	Description
avPar_standbyGet	0	AAAA	Get Standby state
avPar_standbyEnter	1	AAAB	Enter Standby mode
avPar_standbyLeave	2	AAAC	Leave Standby mode - Will cause reboot
Reserved	3	AAAD	
Reserved	4	AAAE	
avPar_sharpness	5	AAAF	Sharpness
avPar_detail	6	AAAG	Detail
avPar_saturation	7	AAAH	Saturation
avPar_hue	8	AAAI	Hue
avPar_pictureFormat	9	AAAJ	Picture Format
Reserved	10	AAAK	
avPar_pixelClockAutofit	11	AAAL	RGBHV auto-fit clock and phase
avPar_pixelClockFreq	12	AAAM	RGBHV auto-fit clock and phase
avPar_pixelClockPhase	13	AAAN	RGBHV Phase adjustment
Reserved	14	AAAO	
Reserved	15	AAAP	
Reserved	16	AAAQ	
Reserved	17	AAAR	
Reserved	18	AAAS	
Reserved	19	AAAT	
Reserved	20	AAAU	
Reserved	21	AAAV	
avPar_inputGamma	22	AAAW	RGB Input Gamma
avPar_inputCapture	23	AAAX	Input Capture / Overscan
avPar_inputWindowPosX	24	AAAY	Input Window Position Horizontal
avPar_inputWindowPosY	25	AAAZ	Input Window Position Vertical
Reserved	26	AAAa	
avPar_outputFormat	27	AAAb	Free-Run/Cross-Lock output standard
Reserved	28	AAAc	
Reserved	29	AAAd	
avPar_outputVidIOLock	30	AAAe	Video I/O Lock (Disabled, Crosslock, Genlock)
avPar_genlockSource	31	AAAf	Genlock Source (Disabled, Component, 3G-SDI)
avPar_genlockStatus	32	AAAg	Genlock Status Info
avPar_outputGammaMode	33	AAAh	RGB Output Gamma

Reserved	34	AAAi	
Reserved	35	AAAj	
Reserved	36	AAAk	
Reserved	37	AAAl	
Reserved	38	AAAm	
Reserved	39	AAAn	
avPar_pan	40	AAAo	Pan
avPar_tilt	41	AAAp	Tilt
avPar_zoomX	42	AAAq	Zoom Horz In/Out
avPar_zoomY	43	AAAr	Zoom Vert In/Out
avPar_panZoomTiltReset	44	AAAs	Reset Pan, Zoom, Tilt
Reserved	45	AAAt	
Reserved	46	AAAu	
Reserved	47	AAAv	
Reserved	48	AAAw	
Reserved	49	AAAx	
Reserved	50	AAAy	
Reserved	51	AAAz	
Reserved	52	AAA0	
avPar_audioDelay	53	AAA1	Audio timeshift
avPar_inVideo	54	AAA2	Select Input
avPar_factoryReset	55	AAA3	Factory Reset
avPar_aspectRatioWidth	56	AAA4	Aspect Ratio width
avPar_aspectRatioHeight	57	AAA5	Aspect Ratio Height
avPar_CCScorrection	58	AAA6	Filter - CCS
avPar_CUEcorrection	59	AAA7	Filter - CUE
avPar_ICPcorrection	60	AAA8	Filter - ICP
avPar_temporalNRLevel	61	AAA9	Temporal Noise Reduction Level
avPar_MPEGNREnable	62	AAA+	MPEG Noise Reduction Enable
avPar_MPEGNRLevel	63	AAA/	MPEG Noise Reduction Level
avPar_movieMode	64	AABA	Movie Mode (auto, video, movie)
avPar_blackLevel	65	AABB	Black level IRE
Reserved	66	AABC	
avPar_inAudio	67	AABD	Current Audio Input
Reserved	68	AABE	
avPar_chTPG_select	69	AABF	Select Test Pattern
avPar_chSVGA_1_audio	70	AABG	Audio Input for RGBHV Input
avPar_chDVI_1_audio	71	AABH	Audio Input for DVI Input
avPar_chHDMI_1_audio	72	AABI	Audio Input for HDMI Input
avPar_chSVid_1_audio	73	AABJ	Audio Input for S-Video 1 Input
Reserved	74	AABK	
avPar_chComp_1_audio	75	AABL	Audio Input for Component 1 Input
Reserved	76	AABM	
avPar_chCVBS_1_audio	77	AABN	Audio Input for CVBS 1 Input
avPar_chCVBS_2_audio	78	AABO	Audio Input for CVBS 2 Input
avPar_chSDI_1_audio	79	AABP	Audio Input for 3G-SDI 1 Input
avPar_chSVGA_1_button	80	AABQ	Input Select Button assigned to Analog Input
avPar_chDVI_1_button	81	AABR	Input Select Button assigned to DVI Input
avPar_chHDMI_1_button	82	AABS	Input Select Button assigned to HDMI Input
avPar_chSVid_1_button	83	AABT	Input Select Button assigned to S-Video 1 Input
Reserved	84	AABU	
avPar_chComp_1_button	85	AABV	Input Select Button assigned to Component 1 Input
Reserved	86	AABW	

avPar_chTPG_button	87	AABX	Input Select Button assigned to Test Pattern Generator
avPar_chCVBS_1_button	88	AAABY	Input Select Button assigned to CVBS 1 Input
avPar_chCVBS_2_button	89	AABZ	Input Select Button assigned to CVBS 2 Input
avPar_chSDI_1_button	90	AABa	Input Select Button assigned to 3G-SDI 1 Input
avPar_chSVGA_1_reset	91	AABb	Reset Analog Input
avPar_chDVI_1_reset	92	AABc	Reset DVI Input
avPar_chHDMI_1_reset	93	AABd	Reset HDMI Input
avPar_chSVid_1_reset	94	AABe	Reset S-Video 1Input
Reserved	95	AABf	
avPar_chComp_1_reset	96	AABg	Reset Component 1 Input
Reserved	97	AABh	
avPar_chTPG_reset	98	AABi	Reset Internal Test Pattern Generator
avPar_chCVBS_1_reset	99	AABj	Reset CVBS 1 Input
avPar_chCVBS_2_reset	100	AABk	Reset CVBS 2 Input
avPar_chSDI_1_reset	101	AABl	Reset 3G-SDI 1 Input
avPar_chComp_1_format	102	AABm	Component 1 Format (RGB, RGBS, YPbPr, 0.7V, 1V etc.)
Reserved	103	AABn	
avPar_colorSpaceIn	104	AABo	Color Space Input (Auto, RGB, YCbCr 4:4:4, YCbCr 4:2:2)
avPar_chHDMI_1_480i	105	AABp	HDMI NTSC (720x480i 60Hz) (off,on)
avPar_chHDMI_1_480p	106	AABq	HDMI NTSC (720x480p 60Hz) (off,on)
avPar_chHDMI_1_576i	107	AABr	HDMI PAL (720x576i 50Hz) (off,on)
avPar_chHDMI_1_576p	108	AABs	HDMI PAL (720x576p 50Hz) (off,on)
avPar_chHDMI_1_720p	109	AABt	HDMI 720p (1280x720) (off,on)
avPar_chHDMI_1_1080i	110	AABu	HDMI 1080p(1920x1080) (off,on)
avPar_chHDMI_1_1080p	111	AABv	HDMI 1080p(1920x1080) (off,on)
Reserved	112	AABw	
avPar_HDMIAudioFormat	113	AABx	HDMI Audio input support
Reserved	114	AABy	
Reserved	115	AABz	
Reserved	116	AAB0	
Reserved	117	AAB1	
Reserved	118	AAB2	
Reserved	119	AAB3	
Reserved	120	AAB4	
Reserved	121	AAB5	
Reserved	122	AAB6	
Reserved	123	AAB7	
Reserved	124	AAB8	
Reserved	125	AAB9	
Reserved	126	AAB+	
Reserved	127	AACA	
Reserved	128	AAB/	
Reserved	129	AACC	
Reserved	130	AACB	
Reserved	131	AACD	
Reserved	132	AACE	
Reserved	133	AACG	
Reserved	134	AACF	
Reserved	135	AACH	
Reserved	136	AACI	
Reserved	137	AACJ	
Reserved	138	AACK	

Reserved	139	AACL	
Reserved	140	AACM	
Reserved	141	AACN	
Reserved	142	AACO	
Reserved	143	AACP	
Reserved	144	AACQ	
Reserved	145	AACR	
Reserved	146	AACS	
Reserved	147	AACT	
Reserved	148	AACU	
Reserved	149	AACV	
Reserved	150	AACW	
Reserved	151	AACY	
Reserved	152	AACX	
Reserved	153	AACZ	
Reserved	154	AACa	
Reserved	155	AACb	
Reserved	156	AACc	
avPar_infoFWVersion	157	AACd	Firmware Version
avPar_infoOutResolution	158	AACe	Output Resolution Info
avPar_infoOutFrameRate	159	AACf	Output Frame Rate Info
avPar_infoInResolution	160	AACh	Input Resolution Info
avPar_infoInFrameRate	161	AACg	Input Frame Rate Info
avPar_infoInHFreq	162	AACi	Input Horizontal Frequency
avPar_infoOutHFreq	163	AACj	Output Horizontal Frequency
avPar_infoGenlock	164	AACk	Genlock Info
avPar_infoGenlockSource	165	AACl	Genlock Source Info
avPar_infoIPAddrSource	166	AACm	IP Address Source Info
avPar_infoDHCPStatus	167	AACn	DHCP Status Info
avPar_infoMACAddress	168	AACo	MAC Address Info
avPar_infoIPAddress	169	AACp	IP Address Info
avPar_infoSubnetMask	170	AACq	Subnet Mask Info
avPar_infoGatewayIP	171	AACr	Gateway IP Address Info
avPar_buttonInSel1_InVid	172	AACs	Input associated with Input Select 1 Button
avPar_buttonInSel2_InVid	173	AACt	Input associated with Input Select 2 Button
avPar_buttonInSel3_InVid	174	AACu	Input associated with Input Select 3 Button
avPar_buttonInSel4_InVid	175	AACv	Input associated with Input Select 4 Button
avPar_buttonInSel1_LED	176	AACw	LED state associated with Input Select 1 Button
avPar_buttonInSel2_LED	177	AACx	LED state associated with Input Select 2 Button
avPar_buttonInSel3_LED	178	AACy	LED state associated with Input Select 3 Button
avPar_buttonInSel4_LED	179	AACz	LED state associated with Input Select 4 Button
avPar_buttonInSel1_select	180	AAC0	Input Select 1 Button
avPar_buttonInSel2_select	181	AAC1	Input Select 2 Button
avPar_buttonInSel3_select	182	AAC2	Input Select 3 Button
avPar_buttonInSel4_select	183	AAC3	Input Select 4 Button
avPar_buttonInput_LED	184	AAC4	LED state associated with Input Button
avPar_buttonInput_select	185	AAC5	Input Button (Cycle through inputs)
avPar_buttonStandby_LED	186	AAC6	LED state associated with Standby Button
avPar_buttonStandby_toggle	187	AAC7	Standby Button (Toggle)

avPar_FPBacklight	188	AAC8	Front Panel Backlight Brightness
Reserved	189	AAC9	
avPar_menuTimeout	190	AAC+	Menu Timeout for Front Panel & OSD
Reserved	191	AAC/	
Reserved	192	AADA	
Reserved	193	AADB	
Reserved	194	AADC	
Reserved	195	AADD	
Reserved	196	AADE	
Reserved	197	AADF	
avPar_outputProcMode	198	AADG	
Reserved	199	AADH	
Reserved	200	AADI	
Reserved	201	AADJ	
Reserved	202	AADK	
Reserved	203	AADL	
Reserved	204	AADM	
Reserved	205	AADN	
Reserved	206	AADO	
Reserved	207	AADP	
Reserved	208	AADQ	
avPar_colorNoSync	209	AADR	No Sync Colour
avPar_preset1Load	210	AADS	Obsolete
avPar_preset2Load	211	AADT	Obsolete
avPar_preset3Load	212	AADU	Obsolete
avPar_preset4Load	213	AADV	Obsolete
avPar_preset1Reset	214	AADW	Preset 1 Reset
avPar_preset2Reset	215	AADX	Preset 2 Reset
avPar_preset3Reset	216	AADY	Preset 3 Reset
avPar_preset4Reset	217	AADZ	Preset 4 Reset
Reserved	218	AADa	
Reserved	219	AADb	
Reserved	220	AADc	
Reserved	221	AADd	
Reserved	222	AADe	
Reserved	223	AADf	
Reserved	224	AADg	
Reserved	225	AADh	
Reserved	226	AADi	
avPar_dhcpEnable	227	AADj	Enable DHCP or static IP Address
avPar_staticIP	228	AADl	Static IP Address
avPar_subnetMask	229	AADk	Static Subnet Mask
avPar_subnetMaskXP	230	AADm	Static Extended Network Prefix displayed as Subnet Mask
avPar_extNetPrefix	231	AADn	Static Extended Network Prefix
avPar_frameRate24Hz	232	AADo	Output Frame Rate when Input is 24Hz (Frame Rate, Output 24Hz, Output 48Hz)
avPar_loadPreset	233	AADp	Select a preset to load
avPar_copyPreset	234	AADq	Copy current settings to preset N
avPar_preset1Name	235	AADr	Preset 1 Name
avPar_preset2Name	236	AADs	Preset 2 Name
avPar_preset3Name	237	AADt	Preset 3 Name
avPar_preset4Name	238	AADu	Preset 4 Name
avPar_currentPreset	239	AADv	Current Preset Info
Reserved	240	AADw	
Reserved	241	AADx	
Reserved	242	AADy	

Reserved	243	AADz	
Reserved	244	AAD0	
Reserved	245	AAD1	
Reserved	246	AAD2	
Reserved	247	AAD3	
Reserved	248	AAD4	
Reserved	249	AAD5	
Reserved	250	AAD6	
Reserved	251	AAD7	
Reserved	252	AAD8	
Reserved	253	AAD9	
Reserved	254	AAD+	
Reserved	255	AAD/	
Reserved	256	AAEA	
Reserved	257	AAEB	
Reserved	258	AAEC	
Reserved	259	AAED	
Reserved	260	AAEE	
Reserved	261	AAEF	
Reserved	262	AAEG	
Reserved	263	AAEH	
Reserved	264	AAEI	
Reserved	265	AAEJ	
Reserved	266	AAEK	
Reserved	267	AAEL	
Reserved	268	AAEM	
avPar_infoBTVersion	269	AAEN	Bootloader Version
Reserved	270	AAEO	
avPar_LegacyMode	271	AAEP	Legacy Mode
avPar_unsharpmask	272	AAEQ	Sharpening Filter
Reserved	273	AAER	
Reserved	274	AAES	
Reserved	275	AAET	
Reserved	276	AAEU	
Reserved	277	AAEV	
Reserved	278	AAEW	
Reserved	279	AAEX	
Reserved	280	AAEY	
Reserved	281	AAEZ	
avPar_VTFilterStrength	282	AAEa	Flicker Filter Strength
avPar_VTFilterRecursive	283	AAEb	Flicker Filter Recursion Level
Reserved	284	AAEc	
Reserved	285	AAEd	
Reserved	286	AAEe	
Reserved	287	AAEf	
Reserved	288	AAEg	
Reserved	289	AAEh	
Reserved	290	AAEi	
avPar_colorSpaceInHDMI	291	AAEj	Input Color Space (Auto, RGB, YCbCr 4:4:4, YCbCr 4:2:2)
avPar_colorSpaceInDVI	292	AAEk	Input Color Space (RGB, YCbCr 4:4:4)
avPar_colorRangeInHDMI	293	AAEl	Input Range (Auto, Full, Limited)
avPar_colorRangeInDVI	294	AAEm	Input Range (Auto, Full, Limited)
avPar_chSDI_2_audio	295	AAEn	Audio Input for 3G-SDI 2 Input
avPar_chSDI_2_button	296	AAEo	Input Select Button assigned to 3G-SDI 2 Input
avPar_chSDI_2_reset	297	AAEp	Input Select Button assigned to 3G-SDI 2 Input

avPar_outputAnalogMode	298	AAEq	Component Video Output mode (YPbPr, RGB)
avPar_output3GSDIMap	299	AAEr	3G-SDI output mode (3G LEVEL_A, 3G LEVEL_B, 10bit_422, 10bit_444_YUV, 10bit_444_RGB, 12bit_444_XYZ)
avPar_lumaGain	300	AAEs	Proc-Amp Luma Gain
avPar_R_Y_Gain	301	AAEt	Proc-Amp Chroma-R Gain
avPar_B_Y_Gain	302	AAEu	Proc-Amp Chroma-B Gain
avPar_lumaOffset	303	AAEv	Proc-Amp Luma Offset
avPar_R_Y_Offset	304	AAEw	Proc-Amp Chroma-R Offset
avPar_B_Y_Offset	305	AAEx	Proc-Amp Chroma-B Offset
avPar_audioChannelSwap01	306	AAEy	Audio Channel Pair Swap
avPar_audioChannelSwap23	307	AAEz	Audio Channel Pair Swap
avPar_audioChannelSwap45	308	AAE0	Audio Channel Pair Swap
avPar_audioChannelSwap67	309	AAE1	Audio Channel Pair Swap
avPar_refVertOffset	310	AAE2	Genlock reference offset V control
avPar_refHorizOffset	311	AAE3	Genlock reference offset H control
Reserved	312	AAE4	
avPar_blueCheck	313	AAE5	Blue check (Off, output Blue only, Green only, Red only, Luma only)
avPar_safeAreaMarkers	314	AAE6	Aspect Ratio fit markers (Off, 4:3, 14:9, 16:9)
avPar_powerConfig	315	AAE7	Dual power supply inlet Monitoring (Off, On)
avPar_infoPowerStatus1	316	AAE8	Power inlet 1 live status
avPar_infoPowerStatus2	317	AAE9	Power inlet 2 live status
avPar_logoOverlayLoc	318	AAE+	Channel Logo overlay location (Off, Top Left, Top Right, Bottom Left, Bottom Right)
avPar_logoARmode	319	AAE/	Channel Logo aspect ratio setting (4:3 safe, 14:9 safe, Widescreen)
avPar_CombFilter	320	AAFA	CVBS 3D Y/C separation (Off, On)
avPar_AFDMode	321	AAFB	Auto Format Detection (Auto, Force)
avPar_AFDefault	322	AAFC	AFD Default/Forced selection (4:3, 16:9, Keep Last)
avPar_AFDSource	323	AAFD	AFD data source (AFD, WSS/CGMS-A, Video Index)
avPar_chSDI_1_stream	324	AAFE	3G-SDI 1 Input Level B Stream (Stream 1, Stream 2)
avPar_chSDI_2_stream	325	AAFF	3G-SDI 2 Input Level B Stream (Stream 1, Stream 2)
avPar_boxReboot	326	AAFG	Force Video Excel reboot
avPar_lockout_enable	327	AAFH	Keypad lockout (Off, On)
avPar_chTPG_speed	328	AAFI	Moving cross test pattern speed
avPar_chTPG_width	329	AAFJ	Moving cross line thickness
avPar_chTPG_fgColour	330	AAFK	Moving cross line colour
avPar_chTPG_bgColour	331	AAFL	Moving cross background colour
avPar_LumaDelay	332	AAFM	Luma/Chroma delay for S-Video
avPar_LumaPeakGain	333	AAFN	Luma peak gain
avPar_CTIGain	334	AAFO	CTI gain
avPar_CTICoringLevel	335	AAFP	CTI coring level
avPar_chCVBS_1_CCLine	336	AAFQ	CC line override for CVBS1 input
avPar_chCVBS_2_CCLine	337	AAFR	CC line override for CVBS2 input
avPar_closedCaptSDIStd	338	AAFS	CEA-608 or 708 CC SD-SDI o/p
avPar_memcFrcStrength	339	AAFT	Level of motion compensation
avPar_chSVid_1_CCLine	340	AAFU	CC line override for S-Video input
avPar_chComp_1_CCLine	341	AAFV	CC line override for Component input

avPar_memcFrcCadenceDetLF	342	AAFW	Cadence detection for 23-30Hz modes
avPar_memcFrcCadenceDetHF	343	AAFX	Cadence detection for 50-60Hz modes
avPar_memcFrcPreserveCad	344	AAFY	Activation of Cadence Preservation
avPar_memcFrcDemoMode	345	AAFZ	Activation of Demo Mode
avPar_memcFrcExclusionZoneMode	346	AAFa	Activation of MEMC Exclusion Zone
avPar_memcFrcExclusionZoneLBorder	347	AAFb	Exclusion Zone left border
avPar_memcFrcExclusionZoneRBorder	348	AAFc	Exclusion Zone right border
avPar_memcFrcExclusionZoneTBorder	349	AAFd	Exclusion Zone top border
avPar_memcFrcExclusionZoneBBorder	350	AAFe	Exclusion Zone bottom border
avPar_memcFrcExclusionZoneSBorder	351	AAFf	Display border
Reserved	352	AAFg	
avPar_memcAdvancedParam1	353	AAFh	Robust SAD Threshold
avPar_memcAdvancedParam2	354	AAFi	SAD Cost Factor
avPar_memcAdvancedParam3	355	AAFj	Temporial/Spatial Balance
avPar_memcAdvancedParam4	356	AAFk	PPC Cost Threshold
avPar_memcAdvancedParam5	357	AAFl	PPC MV Smooth Threshold
avPar_memcAdvancedParam6	358	AAFm	PPC Selection Threshold
avPar_memcAdvancedParam7	359	AAFn	BG/FG Balance Cost
avPar_memcAdvancedParam8	360	AAFo	Huge Motion Distance Threshold
avPar_memcAdvancedParam9	361	AAFp	Huge Motion Fallback Gain
avPar_memcAdvancedParam10	362	AAFq	MV Outlier Threshold
Reserved	363	AAFr	
Reserved	364	AAFs	
avPar_outputTimeCodeEnable	365	AAFt	Enable Timecode output
avPar_TimeCodeOffset	366	AAFu	Time Code offset
avPar_dropframeModeEnable	367	AAFv	Drop frame
avPar_TimeCodeSrcSel	368	AAFw	Time Code source select
avPar_TimeCodeStartVal	369	AAFx	Time Code start value
avPar_TimeCodeTrigVal	370	AAFy	Time Code trigger value
avPar_TimeCodePrerunVal	371	AAFz	Time Code prerun value
avPar_TimeCodeTrigMode	372	AAF0	Trigger mode
avPar_TimeCodeRestart	373	AAF1	Restart Time Code o/p with start value
avPar_TimeCodeInFrMax	374	AAF2	TC maximum frame count on i/p signal
avPar_TimeCodeOutFrMax	375	AAF3	TC max. Frame count on o/p signal
avPar_TimeCodeInFormat	376	AAF4	Embedded LTC/VITC mode on i/p
avPar_TimeCodeOutFormat	377	AAF5	Embedded LTC/VITC mode on o/p
avPar_statusDispMode	378	AAF6	Choose function of status menu
avPar_audioChannelSel03	379	AAF7	Group 1 SDI i/p mapping
avPar_audioChannelSel47	380	AAF8	Group 2 SDI i/p mapping
avPar_audioInChMute01	381	AAF9	Audio i/p 1st stereo channel muting
avPar_audioInChMute23	382	AAF+	Audio i/p 2nd stereo channel muting
avPar_audioInChMute45	383	AAF/	Audio i/p 3rd stereo channel muting
avPar_audioInChMute67	384	AAFA	Audio i/p 4th stereo channel muting
avPar_gatewayIP	385	AAFB	Set gateway IP address

3.2.1.3 Attribute

The attribute identifier specifies the attribute of the parameter this action or request is related to.

Identifier	Value	Base 64	Description
avParamAttr_avail	0	AA	Availability of the parameter.
avParamAttr_availValue	1	AB	Availability of the specific parameter value.
avParamAttr_min	2	AC	Minimum valid parameter value.
avParamAttr_max	3	AD	Maximum valid parameter value.
avParamAttr_value	4	AE	Parameter value (live).
avParamAttr_e2value	5	AF	Parameter value (persistent).

Some attribute are not applicable to some functions as detailed below.

The avParamAttr_value and avParamAttr_e2value attributes allow the separation between the settings in the Audio/Visual engine and the settings in the persistent storage within the unit.

For the get function it is recommended to use the avParamAttr_e2value attribute exclusively.

In the case of the set function the two attribute must be used together as explained in 3.2.5 Set.

For the get string and set string functions there is no difference between the avParamAttr_value and avParamAttr_e2value attributes.

Note avParamAttr_e2value may also be referred to as “NV”, “non-volatile” or “save”.

3.2.1.4 API Debugging (Interpretation)

For the purpose of debugging, if byte 16 is set to an ASCII ‘?’ the unit will return text interpreting what it has received. This looks like this:

```
Func:0, Param:0, Attr:0, Val:00000000
```

The values for Func, Param, Attr are in denary (base 10) and the value for Val is in hexadecimal.

3.2.2 Parameter Reply Packet format

Each function will return one of two packets, one for success and one for an error.

The following table summarises the reply packets for each of the functions.

Function	Function Identifier	Reply Packet Type	Reply Packet Size	Comment
query	avrFunc_query	Success with value	9 bytes	See 3.2.2.2
get	avrFunc_get	Success with value	9 bytes	See 3.2.2.2
set	avrFunc_set	Success	3 bytes	See 3.2.2.1
gets	avrFunc_gets	Success with string	variable	Expect 103 bytes minimum. See 3.2.2.3
sets	avrFunc_sets	Success	3 bytes	See 3.2.2.1
all		Error	15 bytes	See 3.2.2.4

For Ethernet it is necessary to expect the reply to be at least 500 bytes. The reply packet is currently padded to 500 bytes but this is may be removed in the future.

3.2.2.1 Success Reply Packet

For a function that does not return any value the reply packet of a successful call will look like this:

1	2	3
‘O’	‘K’	‘.’

3.2.2.2 Success Reply Packet with return value

For a function that returns a numeric value the reply packet of a successful call will look like this:

1	2	3	4	5	6	7	8	9
‘O’	‘K’	‘:’	value					

Where value is the requested value represented in base 64.

3.2.2.3 Success Reply Packet with string returned

For a function that returns a string the reply packet of a successful call will look like this:

1	2	3	4	… n
‘O’	‘K’	‘:’	ASCII characters …	

The string length will vary. The maximum length of string to expect is 100 characters. Therefore, it is recommended to use a buffer of 103 bytes minimum for receiving this packet.

3.2.2.4 Failed Function (Error) Reply Packet

All function may fail due to errors occurring, in this case an error will be returned.

The reply packet will look like this:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
‘E’	‘R’	‘:’	error						errorLL					

Where

error	is the major error number represented in base 64.
errorLL	is the minor error number represented in base 64.

This table lists the major error codes.

Identifier	**Value**	**Base 64**	**Description**
avParamErr_none	0	AAAAAA	OK, no error
	1	AAAAAB	Undisclosed
avParamErr_param	2	AAAAAC	Parameter invalid or out-of-range.
avParamErr_func	3	AAAAAD	Function is not implemented.
	4	AAAAAE	Undisclosed
	5	AAAAAF	Undisclosed
	6	AAAAAG	Undisclosed
	7	AAAAAH	Undisclosed
	8	AAAAAI	Undisclosed
	9	AAAAAJ	Undisclosed
	10	AAAAAK	Undisclosed
	11	AAAAAL	Undisclosed
avParamErr_remtUnknown	12	AAAAAM	Unknown remote command.
avParamErr_internal	13	AAAAAN	Function failed error in errorLL.

Unlisted or undisclosed major error numbers received suggest a problem with the firmware or an update to this protocol. The minor error number is only useful in debugging firmware and is therefore undisclosed.

3.2.3 Query

The query function provides information about the functions implemented for each parameter. There are two type of query function: single query and a query all functions.

The single query will return true (1) or false (0) indicating if the function for the specified function, attribute, parameter combination is implemented.

The query all will return a value where the binary digits indicate where specific functions/attribute combination as been implemented. Set the function value to avParFunc_queryAll in the following packet.

3.2.3.1 Query Request Packet

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
‘A’	‘P’	‘A’	paramID				attribute		value						‘A’

Where

‘A’, ‘P’	are literal ASCII characters.
paramID	is the parameter identifier being queried.
attribute	is the attribute identifier being queried.
value	is the function being queried (see the following table).

Function to be queried:

Identifier	Value	Base 64	Description
avParFunc_get	0	AAAAAA	Query the get value function.
avParFunc_set	1	AAAAAB	Query the set value function.
avParFunc_gets	2	AAAAAC	Query the get string function.
avParFunc_sets	3	AAAAAD	Query the set string function.
avParFunc_queryAll	4	AAAAAE	Query all the parameter functions.

For query all set attribute to avParamAttr_avail.

3.2.3.2 Query Reply

On successful completion the query function will return a value see “3.2.2.2 Success Reply Packet with return value”.

For a single query function call the possible values returned will be:

0 (AAAAAA in base 64) for false
1 (AAAAAB in base 64) for true

For a query all function call the value return will be a binary bit pattern. Specific binary digits represent true or false indicating if specific functions are implemented.

These are the bit masks to identify each function flag:

Identifier	Bit	Base 16	Description
avQueryFunc_get_avail	0	0x0001	Availability function for parameter.
avQueryFunc_get_availValue	1	0x0002	Availability function for specific value.
avQueryFunc_get_min	2	0x0004	Function that returns the minimum value.
avQueryFunc_get_max	3	0x0008	Function that returns the maximum value.
avQueryFunc_get_val	4	0x0010	Get value function.
avQueryFunc_get_e2val	5	0x0020	Get value function (NV).
avQueryFunc_set_val	6	0x0040	Set value function (live).
avQueryFunc_set_e2val	7	0x0080	Set value function (NV/Save).
avQueryFunc_gets	8	0x0100	Get string function.
avQueryFunc_sets	9	0x0200	Set string function.

3.2.4 Get

The get function is used to obtain a number of attribute of the specified parameter.

The parameter request packet takes the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
‘A’	‘P’	‘B’	paramID				attribute		value						‘A’

Where

‘A’, ‘P’, ‘B’	are literal ASCII characters.
paramID	is the parameter identifier. (see 3.2.1.2 Parameter Identifier)
attribute	is the attribute identifier. (see 3.2.1.3 Attribute)
value	is a value.

The possible attributes that can be obtained are: parameter availability, value availability, minimum value, maximum value, parameter value (live) and parameter value (non-volatile/save).

3.2.4.1 Parameter Availability

The parameter availability function is a used to determine if this parameter is currently available to be read or changed.

Attribute should be set to avParamAttr_avail from the table in 3.2.1.3 Attribute. Value is not used and should be set to zero.

A successful call will return true (1) or false (0). See “3.2.2.2 Success Reply Packet with return value”. True (1) indicates the parameter may be read or changed using any of the implemented functions.

3.2.4.2 Parameter Value Availability

The parameter value availability function is a used to determine if the specified value is currently a valid value for this particular parameter.

Attribute should be set to avParamAttr_availValue from the table in 3.2.1.3 Attribute. Value specifies the value to be queried.

A successful call will return true (1) or false (0). See “3.2.2.2 Success Reply Packet with return value”. True (1) indicates the parameter value may be set using the set function if implemented.

3.2.4.3 Minimum and Maximum

The parameter minimum and maximum function is a used to obtain the minimum or maximum value of the parameters’ range of values.

Attribute should be set to avParamAttr_min or avParamAttr_max from the table in 3.2.1.3 Attribute. Value is not used and should be set to zero.

A successful call will return the relevant value. See “3.2.2.2 Success Reply Packet with return value”.

3.2.4.4 Parameter Value

The parameter value may be obtained using the attributes avParamAttr_value or avParamAttr_e2value. The two value attributes represent the live setting value and the value stored in persistent memory respectively.

It is recommended to use the avParamAttr_e2value attribute exclusively to obtain the current value.

Value is not used and should be set to zero.

The returned value is the value of the parameter setting. See “3.2.2.2 Success Reply Packet with return value”.

3.2.5 Set

The set function is used to change the setting of the specified parameter.

The parameter request packet takes the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
'A'	'P'	'C'	paramID				attribute		value						'A'

Where

'A', 'P', 'C'	are literal ASCII characters.
paramID	is the parameter identifier. (see 3.2.1.2 Parameter Identifier)
attribute	is the attribute identifier. (see 3.2.1.3 Attribute)
value	is a value.

Only the avParamAttr_value and avParamAttr_e2value attributes may be set and these must be use together. The avParamAttr_value attribute changes the specified parameters' value. The avParamAttr_e2value attribute either writes the specified value into persistent storage or writes the value that was previously set using the avParamAttr_value attribute. Writing to persistent storage can be slow therefore the recommended usage is to use the avParamAttr_value attribute for interactive adjustment and then complete the adjustment with one set of the avParamAttr_e2value attribute.

It is advisable follow the example scenario presented in this document (see 3.3.5.3.1) when creating user interfaces using the protocol.

Only the values between the minimum and maximum values (obtained using the get function) are valid.

The set function on successful completion will return a successful reply packet. See "3.2.2.1 Success Reply Packet".

3.2.6 Get String

The get string function is used to obtain a string value. This may be used to convert the numeric value into a string or it may be used to obtain information in the form of a string (in this case the get value function will not be implemented).

The parameter request packet takes the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
'A'	'P'	'D'	paramID				attribute		Value						'A'

Where

'A', 'P', 'D'	are literal ASCII characters.
paramID	is the parameter identifier. (see 3.2.1.2 Parameter Identifier)
attribute	is the attribute identifier. (see 3.2.1.3 Attribute)
value	is a value.

There is no difference between the avParamAttr_value and avParamAttr_e2value attributes. The current recommendation is to set attribute to avParamAttr_value.

For the reply packet see "3.2.2.3 Success Reply Packet with string returned".

3.2.7 Set String

The set string function is used to set a string, however it is restricted to six ASCII characters. Only a few parameters implement this function.

The parameter request packet takes the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
‘A’	‘P’	‘E’	paramID				attribute		Value						‘A’

Where

‘A’, ‘P’, ‘E’	are literal ASCII characters.
paramID	is the parameter identifier. (see 3.2.1.2 Parameter Identifier)
attribute	is the attribute identifier. (see 3.2.1.3 Attribute)
value	is a value.

There is no difference between the avParamAttr_value and avParamAttr_e2value attributes. The current recommendation is to set attribute to avParamAttr_value.

The string must be six ASCII characters long. If it is shorter the space character (‘ ’, 32, 0x20) must be used to pad the string. Put this string in the Value bytes. Null termination should NOT be used.

The set string function on successful completion will return a successful reply packet. See “3.2.2.1 Success Reply Packet”.

3.3 **Guidelines for AV API Usage**

The AV API was designed to be a generalised interface to control units and as such provides access to a number of different parameter types. This is reflected in the functions that are implemented for the various parameters. It is advisable to follow these guidelines when implementing any user interface for controlling units.

3.3.1 Query Function

The purpose of query function is to ascertain which functions are available to a parameter. This gives an idea as to what type parameter is. Secondly, the query function may improve performance using the interface. For function that are not implemented there is little point in calling that function. This is particularly important with the availability functions.

3.3.2 Parameter Availability Function

If the availability function for a parameter is implemented it should be called prior to using the other function relating to that parameter. If the availability function returns false (0) the user interface should indicate by some means that this parameter is not available. Note that parameters may change their availability status based changes to other parameters or the input signal etc.

3.3.3 Parameter Value Range

For parameters that map to a number (integer) the minimum and maximum values should be obtained. These should be used by the user interface to restrict the user input to valid values. If the minimum and maximum function are not implemented the parameter is either not a numeric parameter or is read only.

3.3.4 Parameter Value Availability Function

The value availability function is used to restrict even further the valid values that may be set. When implemented the value availability function should be called to verify the validity of a value prior to calling the set functions.
If the value availability function is not implemented all values between the minimum and maximum values inclusive are valid.

Note that parameter values may change their availability status based changes to other parameters or the input signal etc.

3.3.5 Parameter Types

The different parameter types reflect the distinct usages of the API. For example, the picture contrast parameter is a number (integer) and therefore requires functionality to allow the contrast value to be read and changed. Whereas, a factory reset is an action and therefore only requires a means to trigger that action. Some parameters are used to provide information and implement only the get string function.

3.3.5.1 Parameter type: Info

Parameters that conform to the info type only implement the get string function. These are parameters that provide status information in the form of a string, as is.
E.g. avPar_infoOutResolution, avPar_infoOutFrameRate, avPar_infoInResolution.

3.3.5.2 Parameter type: Action

Action parameters are used to trigger actions such as factory reset and channel resets. To perform the action the set function is used with attribute set to avParamAttr_value and value is set to zero. The availability function should be used prior, to ascertain if the action is available.

3.3.5.3 Parameter type: Integer

The majority of parameters are numeric and are classed as integers. Some parameters that do not appear to be numeric to the user are also of integer type. The get string function is used to convert the numeric value into a string and hides the numbers from the user. Note that the get string function may return a different string for a particular value on separate occasions.

3.3.5.3.1 Example scenario

To demonstrate how all the functions are interrelated and work together, the following sections present pseudo code showing how the API is intended to be used and is used within our own user interface software both in firmware and on a PC.

3.3.5.3.2 Connect to unit

The query function needs only be call once per connection.

```
boolean succ;
int qfuncs;

succ = queryAll(paramID => avPar_contrast, result => qfuncs);
if succ then
   test qfuncs for min, max, value, e2val
else
   Process Error
end if
```

3.3.5.3.3 Availability Check

The availability function needs to be tested before editing a parameter.

```
boolean succ;
int avail;

succ = checkAvail(paramID => avPar_contrast, result => avail);
if succ then
          if avail then
             activate UI control
          else
             display not available or grey out GUI control
          end if
else
   Process Error
end if
```

3.3.5.3.4 Initiating UI Control for Editing

Having established the parameter is available and the user selects the parameter for editing the user interface needs to obtain the range and current value.

```
int qfuncs;      -- obtained on connection
boolean succ;
int min;
int max;
int val;
int valAvail;
string st;

succ = get(paramID => avPar_contrast,
        attribute => avParamAttr_min,
        value => min);
if succ then
   succ = get(paramID => avPar_contrast,
           attribute => avParamAttr_max,
           value => max);
end if
if succ then
```

```
   succ = get(paramID => avPar_contrast,
              attribute => avParamAttr_e2value,
              value => val);
end if
if succ then
   if (qfuncs and avQueryFunc_get_availValue) > 0 then
      valAvail = val;
      succ = get(paramID => avPar_contrast,
                 attribute => avParamAttr_availValue,
                 value => valAvail);
   else
      valAvail = TRUE;
   end if
end if
if succ then
   if (qfuncs and avQueryFunc_gets) > 0 then
      succ = gets(paramID => avPar_contrast,
                  attribute => avParamAttr_value,
                  value => val,
                  returned string => st);
   else
      st = convertToString(val);
   end if

   set UI control range to min, max
   display st as value in UI control
   if not valAvail then
      indicate on UI this particular value is not available
   end if
   allow user to use control
else
   Process Error
end if
```

3.3.5.3.5 The User Adjusts the Value

Having allowed the user to make an adjustment to the value on the user interface, the unit needs to be updated. Here the set is only performed with the avParamAttr_value attribute. This is for performance reasons. It must also be set with using the avParamAttr_e2value attribute when the user as finished (as will be shown in the next section).

```
boolean succ;
int qfuncs;    -- obtained on connection
int min;       -- obtained on start of editing
int max;       -- obtained on start of editing
int newval;    -- new adjusted value
int oldval;    -- unadjusted value currently on unit
int valAvail;

if (newval >= min) and (newval <= max) then
   if (qfuncs and avQueryFunc_get_availValue) > 0 then
      valAvail = newval;
      succ = get(paramID => avPar_contrast,
                 attribute => avParamAttr_availValue,
                 value => valAvail);
   else
      valAvail = TRUE;
   end if
   if valAvail then
      succ = set(paramID => avPar_contrast,
                 attribute => avParamAttr_value,
                value => newval);
        if succ then
            oldval = newval; -- update old value
        end if
    else
        -- value not available: do not set
        indicate on UI that value is not available
    end if
else
    newval = oldval; -- revert change
end if
if succ then
    if (qfuncs and avQueryFunc_gets) > 0 then
        succ = gets(paramID => avPar_contrast,
                attribute => avParamAttr_value,
                value => val,
                returned string => st);
    else
        st = convertToString(val);
    end if

    display st as value in UI control
    if not valAvail then
        indicate on UI this particular value is not available
    end if
    allow user to use control
else
    Process Error
end if
```

3.3.5.3.6 The User Completes Adjusting the Value

When the user has finished adjusting the parameter it is necessary to set the value using both of the value attributes.

```
boolean succ;
int oldval;    -- unadjusted value currently on unit
int valAvail;

succ = set(paramID => avPar_contrast,
        attribute => avParamAttr_value,
        value => oldval);
if succ then
    succ = set(paramID => avPar_contrast,
            attribute => avParamAttr_e2value,
            value => oldval);
end if
```

3.3.5.4 Parameter type: String

String parameters implement both the get string and set string functions. The minimum and maximum functions when implemented provide the minimum and maximum length of the string. The protocol has a restriction of six characters maximum for the set string function; this is a limitation of this protocol and the underlying communication implementation.

3.4 Base 64 Encoding

Base 64 is used as a compromise between compressing numeric values whilst still using human readable ASCII characters. Base 64 is used by various internet protocols.

Base 64 is defined in “RFC 3548 The Base16, Base32, and Base64 Data Encodings July 2003” and is available on the internet at http://www.faqs.org/rfcs/rfc3548.html.

The digits of base 64 are listed in the following table:

Table 1: The Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

The packets of this protocol have fixed length field for numeric data. They must be padded with zeros to fully occupy the allotted bytes. From the above table it is noted that in base 64 the zero is the ASCII upper case ‘A’. In the section 3.4.1 Base 64 Ready Reckoner example conversion of number may be obtained to assist in creating relevant conversion functions.

3.4.1 Base 64 Ready Reckoner

Here is a ready table showing the conversion between denary/decimal (base 10), hexadecimal (base 16) and base 64 for the numbers between zero and five hundred. Base 64 numbers are padded to six characters. For shorter fields remove the leading ‘A’s.

Base 10	Base 16	Base 64
0	0x000	AAAAAA
1	0x001	AAAAAB
2	0x002	AAAAAC
3	0x003	AAAAAD
4	0x004	AAAAAE
5	0x005	AAAAAF
6	0x006	AAAAAG
7	0x007	AAAAAH
8	0x008	AAAAAI
9	0x009	AAAAAJ
10	0x00A	AAAAAK
11	0x00B	AAAAAL
12	0x00C	AAAAAM
13	0x00D	AAAAAN
14	0x00E	AAAAAO
15	0x00F	AAAAAP
16	0x010	AAAAAQ
17	0x011	AAAAAR
18	0x012	AAAAAS
19	0x013	AAAAAT
20	0x014	AAAAAU
21	0x015	AAAAAV
22	0x016	AAAAAW
23	0x017	AAAAAX
24	0x018	AAAAAY
25	0x019	AAAAAZ
26	0x01A	AAAAAa
27	0x01B	AAAAAb
28	0x01C	AAAAAc
29	0x01D	AAAAAd
30	0x01E	AAAAAe
31	0x01F	AAAAAf
32	0x020	AAAAAg
33	0x021	AAAAAh
34	0x022	AAAAAi
35	0x023	AAAAAj
36	0x024	AAAAAk
37	0x025	AAAAAl
38	0x026	AAAAAm
39	0x027	AAAAAn
40	0x028	AAAAAo
41	0x029	AAAAAp
42	0x02A	AAAAAq
43	0x02B	AAAAAr
44	0x02C	AAAAAs
45	0x02D	AAAAAt
46	0x02E	AAAAAu
47	0x02F	AAAAAv
48	0x030	AAAAAw
49	0x031	AAAAAx
50	0x032	AAAAAy
51	0x033	AAAAAz
52	0x034	AAAAA0
53	0x035	AAAAA1
54	0x036	AAAAA2
55	0x037	AAAAA3
56	0x038	AAAAA4
57	0x039	AAAAA5
58	0x03A	AAAAA6
59	0x03B	AAAAA7
60	0x03C	AAAAA8
61	0x03D	AAAAA9
62	0x03E	AAAAA+
63	0x03F	AAAAA/
64	0x040	AAAABA
65	0x041	AAAABB
66	0x042	AAAABC
67	0x043	AAAABD
68	0x044	AAAABE
69	0x045	AAAABF
70	0x046	AAAABG
71	0x047	AAAABH
72	0x048	AAAABI
73	0x049	AAAABJ
74	0x04A	AAAABK
75	0x04B	AAAABL
76	0x04C	AAAABM
77	0x04D	AAAABN
78	0x04E	AAAABO
79	0x04F	AAAABP
80	0x050	AAAABQ
81	0x051	AAAABR
82	0x052	AAAABS
83	0x053	AAAABT
84	0x054	AAAABU
85	0x055	AAAABV
86	0x056	AAAABW
87	0x057	AAAABX
88	0x058	AAAABY
89	0x059	AAAABZ
90	0x05A	AAAABa
91	0x05B	AAAABb
92	0x05C	AAAABc
93	0x05D	AAAABd
94	0x05E	AAAABe
95	0x05F	AAAABf
96	0x060	AAAABg
97	0x061	AAAABh
98	0x062	AAAABi
99	0x063	AAAABj
100	0x064	AAAABk
101	0x065	AAAABl
102	0x066	AAAABm
103	0x067	AAAABn
104	0x068	AAAABo
105	0x069	AAAABp
106	0x06A	AAAABq
107	0x06B	AAAABr
108	0x06C	AAAABs
109	0x06D	AAAABt
110	0x06E	AAAABu
111	0x06F	AAAABv
112	0x070	AAAABw
113	0x071	AAAABx
114	0x072	AAAABy
115	0x073	AAAABz
116	0x074	AAAAB0
117	0x075	AAAAB1
118	0x076	AAAAB2
119	0x077	AAAAB3
120	0x078	AAAAB4
121	0x079	AAAAB5
122	0x07A	AAAAB6
123	0x07B	AAAAB7
124	0x07C	AAAAB8
125	0x07D	AAAAB9
126	0x07E	AAAAB+
128	0x080	AAAACA
127	0x07F	AAAAB/
130	0x082	AAAACC
129	0x081	AAAACB
131	0x083	AAAACD
132	0x084	AAAACE
134	0x086	AAAACG
133	0x085	AAAACF
135	0x087	AAAACH
136	0x088	AAAACI
137	0x089	AAAACJ
138	0x08A	AAAACK
139	0x08B	AAAACL
140	0x08C	AAAACM
141	0x08D	AAAACN
142	0x08E	AAAACO
143	0x08F	AAAACP
144	0x090	AAAACQ
145	0x091	AAAACR

146	0x092	AAAACS
147	0x093	AAAACT
148	0x094	AAAACU
149	0x095	AAAACV
150	0x096	AAAACW
152	0x098	AAAACY
151	0x097	AAAACX
153	0x099	AAAACZ
154	0x09A	AAAACa
155	0x09B	AAAACb
156	0x09C	AAAACc
157	0x09D	AAAACd
158	0x09E	AAAACe
159	0x09F	AAAACf
161	0x0A1	AAAACh
160	0x0A0	AAAACg
162	0x0A2	AAAACi
163	0x0A3	AAAACj
164	0x0A4	AAAACk
165	0x0A5	AAAACl
166	0x0A6	AAAACm
167	0x0A7	AAAACn
168	0x0A8	AAAACo
169	0x0A9	AAAACp
170	0x0AA	AAAACq
171	0x0AB	AAAACr
172	0x0AC	AAAACs
173	0x0AD	AAAACt
174	0x0AE	AAAACu
175	0x0AF	AAAACv
176	0x0B0	AAAACw
177	0x0B1	AAAACx
178	0x0B2	AAAACy
179	0x0B3	AAAACz
180	0x0B4	AAAAC0
181	0x0B5	AAAAC1
182	0x0B6	AAAAC2
183	0x0B7	AAAAC3
184	0x0B8	AAAAC4
185	0x0B9	AAAAC5
186	0x0BA	AAAAC6
187	0x0BB	AAAAC7
188	0x0BC	AAAAC8
189	0x0BD	AAAAC9
190	0x0BE	AAAAC+
191	0x0BF	AAAAC/
192	0x0C0	AAAADA
193	0x0C1	AAAADB
194	0x0C2	AAAADC
195	0x0C3	AAAADD
196	0x0C4	AAAADE
197	0x0C5	AAAADF
198	0x0C6	AAAADG
199	0x0C7	AAAADH
200	0x0C8	AAAADI
201	0x0C9	AAAADJ
202	0x0CA	AAAADK
203	0x0CB	AAAADL
204	0x0CC	AAAADM
205	0x0CD	AAAADN
206	0x0CE	AAAADO
207	0x0CF	AAAADP
208	0x0D0	AAAADQ
209	0x0D1	AAAADR
210	0x0D2	AAAADS
211	0x0D3	AAAADT
212	0x0D4	AAAADU
213	0x0D5	AAAADV
214	0x0D6	AAAADW
215	0x0D7	AAAADX
216	0x0D8	AAAADY
217	0x0D9	AAAADZ
218	0x0DA	AAAADa
219	0x0DB	AAAADb
220	0x0DC	AAAADc
221	0x0DD	AAAADd
222	0x0DE	AAAADe
223	0x0DF	AAAADf
224	0x0E0	AAAADg
225	0x0E1	AAAADh
226	0x0E2	AAAADi
227	0x0E3	AAAADj
229	0x0E5	AAAADl
228	0x0E4	AAAADk
230	0x0E6	AAAADm
231	0x0E7	AAAADn
232	0x0E8	AAAADo
233	0x0E9	AAAADp
234	0x0EA	AAAADq
235	0x0EB	AAAADr
236	0x0EC	AAAADs
237	0x0ED	AAAADt
238	0x0EE	AAAADu
239	0x0EF	AAAADv
240	0x0F0	AAAADw
241	0x0F1	AAAADx
242	0x0F2	AAAADy
243	0x0F3	AAAADz
244	0x0F4	AAAAD0
245	0x0F5	AAAAD1
246	0x0F6	AAAAD2
247	0x0F7	AAAAD3
248	0x0F8	AAAAD4
249	0x0F9	AAAAD5
250	0x0FA	AAAAD6
251	0x0FB	AAAAD7
252	0x0FC	AAAAD8
253	0x0FD	AAAAD9
254	0x0FE	AAAAD+
255	0x0FF	AAAAD/
256	0x100	AAAAEA
257	0x101	AAAAEB
258	0x102	AAAAEC
259	0x103	AAAAED
260	0x104	AAAAEE
261	0x105	AAAAEF
262	0x106	AAAAEG
263	0x107	AAAAEH
264	0x108	AAAAEI
265	0x109	AAAAEJ
266	0x10A	AAAAEK
267	0x10B	AAAAEL
268	0x10C	AAAAEM
269	0x10D	AAAAEN
270	0x10E	AAAAEO
271	0x10F	AAAAEP
272	0x110	AAAAEQ
273	0x111	AAAAER
274	0x112	AAAAES
275	0x113	AAAAET
276	0x114	AAAAEU
277	0x115	AAAAEV
278	0x116	AAAAEW
279	0x117	AAAAEX
280	0x118	AAAAEY
281	0x119	AAAAEZ
282	0x11A	AAAAEa
283	0x11B	AAAAEb
284	0x11C	AAAAEc
285	0x11D	AAAAEd
286	0x11E	AAAAEe
287	0x11F	AAAAEf
288	0x120	AAAAEg
289	0x121	AAAAEh
290	0x122	AAAAEi
291	0x123	AAAAEj
292	0x124	AAAAEk

293	0x125	AAAAEl
294	0x126	AAAAEm
295	0x127	AAAAEn
296	0x128	AAAAEo
297	0x129	AAAAEp
298	0x12A	AAAAEq
299	0x12B	AAAAEr
300	0x12C	AAAAEs
301	0x12D	AAAAEt
302	0x12E	AAAAEu
303	0x12F	AAAAEv
304	0x130	AAAAEw
305	0x131	AAAAEx
306	0x132	AAAAEy
307	0x133	AAAAEz
308	0x134	AAAAE0
309	0x135	AAAAE1
310	0x136	AAAAE2
311	0x137	AAAAE3
312	0x138	AAAAE4
313	0x139	AAAAE5
314	0x13A	AAAAE6
315	0x13B	AAAAE7
316	0x13C	AAAAE8
317	0x13D	AAAAE9
318	0x13E	AAAAE+
319	0x13F	AAAAE/
320	0x140	AAAAFA
321	0x141	AAAAFB
322	0x142	AAAAFC
323	0x143	AAAAFD
324	0x144	AAAAFE
325	0x145	AAAAFF
326	0x146	AAAAFG
327	0x147	AAAAFH
328	0x148	AAAAFI
329	0x149	AAAAFJ
330	0x14A	AAAAFK
331	0x14B	AAAAFL
332	0x14C	AAAAFM
333	0x14D	AAAAFN
334	0x14E	AAAAFO
335	0x14F	AAAAFP
336	0x150	AAAAFQ
337	0x151	AAAAFR
338	0x152	AAAAFS
339	0x153	AAAAFT
340	0x154	AAAAFU
341	0x155	AAAAFV

342	0x156	AAAAFW
343	0x157	AAAAFX
344	0x158	AAAAFY
345	0x159	AAAAFZ
346	0x15A	AAAAFa
347	0x15B	AAAAFb
348	0x15C	AAAAFc
349	0x15D	AAAAFd
350	0x15E	AAAAFe
351	0x15F	AAAAFf
352	0x160	AAAAFg
353	0x161	AAAAFh
354	0x162	AAAAFi
355	0x163	AAAAFj
356	0x164	AAAAFk
357	0x165	AAAAFl
358	0x166	AAAAFm
359	0x167	AAAAFn
360	0x168	AAAAFo
361	0x169	AAAAFp
362	0x16A	AAAAFq
363	0x16B	AAAAFr
364	0x16C	AAAAFs
365	0x16D	AAAAFt
366	0x16E	AAAAFu
367	0x16F	AAAAFv
368	0x170	AAAAFw
369	0x171	AAAAFx
370	0x172	AAAAFy
371	0x173	AAAAFz
372	0x174	AAAAF0
373	0x175	AAAAF1
374	0x176	AAAAF2
375	0x177	AAAAF3
376	0x178	AAAAF4
377	0x179	AAAAF5
378	0x17A	AAAAF6
379	0x17B	AAAAF7
380	0x17C	AAAAF8
381	0x17D	AAAAF9
382	0x17E	AAAAF+
383	0x17F	AAAAF/
384	0x180	AAAAGA
385	0x181	AAAAGB
386	0x182	AAAAGC
387	0x183	AAAAGD
388	0x184	AAAAGE
389	0x185	AAAAGF
390	0x186	AAAAGG

391	0x187	AAAAGH
392	0x188	AAAAGI
393	0x189	AAAAGJ
394	0x18A	AAAAGK
395	0x18B	AAAAGL
396	0x18C	AAAAGM
398	0x18E	AAAAGO
397	0x18D	AAAAGN
399	0x18F	AAAAGP
400	0x190	AAAAGQ
401	0x191	AAAAGR
402	0x192	AAAAGS
403	0x193	AAAAGT
404	0x194	AAAAGU
405	0x195	AAAAGV
406	0x196	AAAAGW
407	0x197	AAAAGX
408	0x198	AAAAGY
409	0x199	AAAAGZ
410	0x19A	AAAAGa
411	0x19B	AAAAGb
412	0x19C	AAAAGc
413	0x19D	AAAAGd
414	0x19E	AAAAGe
415	0x19F	AAAAGf
416	0x1A0	AAAAGg
417	0x1A1	AAAAGh
418	0x1A2	AAAAGi
419	0x1A3	AAAAGj
420	0x1A4	AAAAGk
421	0x1A5	AAAAGl
422	0x1A6	AAAAGm
423	0x1A7	AAAAGn
424	0x1A8	AAAAGo
425	0x1A9	AAAAGp
426	0x1AA	AAAAGq
427	0x1AB	AAAAGr
428	0x1AC	AAAAGs
429	0x1AD	AAAAGt
430	0x1AE	AAAAGu
431	0x1AF	AAAAGv
432	0x1B0	AAAAGw
434	0x1B2	AAAAGy
433	0x1B1	AAAAGx
435	0x1B3	AAAAGz
436	0x1B4	AAAAG0
437	0x1B5	AAAAG1
438	0x1B6	AAAAG2
439	0x1B7	AAAAG3

440	0x1B8	AAAAG4
441	0x1B9	AAAAG5
442	0x1BA	AAAAG6
443	0x1BB	AAAAG7
444	0x1BC	AAAAG8
445	0x1BD	AAAAG9
446	0x1BE	AAAAG+
447	0x1BF	AAAAG/
448	0x1C0	AAAAHA
449	0x1C1	AAAAHB
450	0x1C2	AAAAHC
451	0x1C3	AAAAHD
452	0x1C4	AAAAHE
453	0x1C5	AAAAHF
454	0x1C6	AAAAHG
455	0x1C7	AAAAHH
456	0x1C8	AAAAHI
457	0x1C9	AAAAHJ
458	0x1CA	AAAAHK
459	0x1CB	AAAAHL
460	0x1CC	AAAAHM

461	0x1CD	AAAAHN
462	0x1CE	AAAAHO
463	0x1CF	AAAAHP
464	0x1D0	AAAAHQ
465	0x1D1	AAAAHR
466	0x1D2	AAAAHS
467	0x1D3	AAAAHT
468	0x1D4	AAAAHU
469	0x1D5	AAAAHV
470	0x1D6	AAAAHW
471	0x1D7	AAAAHX
472	0x1D8	AAAAHY
473	0x1D9	AAAAHZ
474	0x1DA	AAAAHa
475	0x1DB	AAAAHb
476	0x1DC	AAAAHc
477	0x1DD	AAAAHd
478	0x1DE	AAAAHe
479	0x1DF	AAAAHf
480	0x1E0	AAAAHg
481	0x1E1	AAAAHh

482	0x1E2	AAAAHi
483	0x1E3	AAAAHj
484	0x1E4	AAAAHk
485	0x1E5	AAAAHl
486	0x1E6	AAAAHm
487	0x1E7	AAAAHn
488	0x1E8	AAAAHo
489	0x1E9	AAAAHp
490	0x1EA	AAAAHq
491	0x1EB	AAAAHr
492	0x1EC	AAAAHs
494	0x1EE	AAAAHu
493	0x1ED	AAAAHt
495	0x1EF	AAAAHv
496	0x1F0	AAAAHw
497	0x1F1	AAAAHx
498	0x1F2	AAAAHy
499	0x1F3	AAAAHz
500	0x1F4	AAAAH0